VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2016 Ondřej Pospíchal

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

MONITOROVÁNÍ SERVERŮ V INTERNETU
MONITORING OF INTERNET SERVERS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Ondřej Pospíchal**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Dan Komosný, Ph.D.**
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Ondřej Pospíchal ***ID:*** 164370
Ročník: 3 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Monitorování serverů v Internetu

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s experimentální sítí PlanetLab – http://www.planet-lab.org/. Vytvořte seznam stanic sítě PlanetLab, které se nacházejí v Evropě. Zhotovte aplikaci v OS Linux, která bude pravidelně testovat dostupnost těchto serverů pomocí vzdáleného připojení SSH. Výsledky ukládejte do textového souboru v předepsaném formátu.

DOPORUČENÁ LITERATURA:

[1] PUŽMANOVÁ, R. TCP/IP v kostce. 2. vyd. Kopp, 2009. 620 s. ISBN: 978-80-7232-388-3.

[2] Linux Dokumentační projekt. 4. vyd. Computer Press, 2008. 1336 s. ISBN: 978-80-251-1525-1.

[3] CASTRO. E. HTML, XHTML a CSS :názorný průvodce tvorbou WWW stránek. 1. vyd. Computer Press, 2007. 440 s. ISBN: 978-80-251-1531-2.

Termín zadání: 1.2.2016 ***Termín odevzdání:*** 1.6.2016

Vedoucí práce: doc. Ing. Dan Komosný, Ph.D.
Konzultant bakalářské práce:

doc. Ing. Jiří Mišurec, CSc., *předseda oborové rady*

ABSTRAKT

Tato práce se zabývá získáváním pravidelných informací o dostupnosti stanic a o dostupnosti vzdáleného připojení stanic pomocí SSH v síti PlanetLab po celém světě. V teoretické části je popsaná síť PlanetLab, služby ping a SSH, programové prostředí, ve kterém vznikala praktická část práce, a paralelizace procesů, která byla využita při optimalizaci programu. Praktická část popisuje tvorbu, funkce a výstupy programu, který nepřetržitě testuje servery sítě PlanetLab. Analýzou výsledků programu bylo zjištěno, že dostupnost zařízení PlanetLabu je poměrně stálá, ale neuspokojivá. Více než polovina zařízení této sítě totiž neodpovídá ani na jednu z uvedených služeb.

KLÍČOVÁ SLOVA

PlanetLab, ping, SSH, monitorování, testování, skript, Bash

ABSTRACT

This thesis deals with the availability of devices and the availability of remote connection to devices via SSH in the network PlanetLab worldwide. The theoretical part consists of the description of the network PlanetLab, services ping and SSH, program environment, in which was the practical part made, and parallelization of processes, which was used during optimizing the program. The practical part describes the creation, functions and outputs of the application, which continuously tests servers in the network PlanetLab. Analyzing the results of the program it was found that the availability of PlanetLab devices is relatively stable, but unsatisfactory. More than half of the devices of this network do not respond to neither one of mentioned services.

KEYWORDS

PlanetLab, ping, SSH, monitoring, testing, script, Bash

POSPÍCHAL, Ondřej *Monitorování serverů v Internetu*: bakalářská práce. BRNO: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 43 s. Vedoucí práce byl doc. Ing. Dan Komosný, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Monitorování serverů v Internetu“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

BRNO ....

podpis autora(-ky)

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce doc. Ing. Danu Komosnému, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

BRNO ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

BRNO ....

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

Úvod **11**

1 Experimentální síť PlanetLab **12**
1.1 Terminologie 12
1.2 Historie sítě PlanetLab 12
1.3 Projekty v síti PlanetLab 13

2 Služby ping a SSH **15**
2.1 Služba ping 15
2.2 Služba SSH 15

3 Programové prostředí **18**

4 Paralelizace procesů **20**
4.1 Typy paralelizace 20
4.2 Postup při návrhu paralelizace 21

5 Vytvořený program **22**
5.1 Postup tvorby skriptu a jeho funkce 22
5.2 Výstup skriptu. 28
5.3 Pravidelné spouštění skriptu 29
5.4 Umístění skriptu na server 30

6 Analýza naměřených dat **32**
6.1 Dostupnost služeb ping a SSH na zařízeních sítě PlanetLab 32
6.2 Dlouhodobá statistika dostupnosti služeb v PlanetLabu 33
6.3 Ověření funkčnosti paralelizace 37

7 Závěr **38**

Literatura **39**

Seznam symbolů, veličin a zkratek **41**

Seznam příloh **42**

A Obsah přiloženého CD **43**

SEZNAM OBRÁZKŮ

5.1 Vývojový diagram aplikace pro monitorování serverů v Internetu. . . 23
6.1 Měření dostupnosti služeb 3.4.2016. 32
6.2 Graf počtu zařízení dostupných zároveň na ping i SSH. 34
6.3 Graf počtu zařízení dostupných pouze na ping. 34
6.4 Graf počtu zařízení dostupných pouze na SSH. 35
6.5 Graf počtu nedostupných zařízení. 36
6.6 Graf počtu všech zařízení dostupných na ping, na SSH a nedostupných. 36

SEZNAM VÝPISŮ KÓDU

5.1 Metoda Monitoring. 25
5.2 Cyklus spouštějící metodu Monitoring. 26
5.3 Výpočet statistik. 27
5.4 Část výstupního textového souboru *Result.txt*. 28
5.5 Výstupní datový soubor *metadata.dat*. 29

SEZNAM TABULEK

6.1 Tabulka ušetřeného času pomocí paralelizace. 37

ÚVOD

Cílem této práce je seznámení se s experimentální sítí PlanetLab a vytvoření aplikace v operačním systému Linux, která bude pravidelně monitorovat dostupnost serverů zmíněné experimentální sítě na služby ping a SSH. Výsledky jednotlivých testování bude aplikace ukládat do textových souborů. Tato práce tvoří vetší celek dohromady s prací „Výzkumná síť PlanetLab“, jejíž autorem je Matej Grejták. Aplikace Matěje Grejtáka pracuje s výsledky této práce a pravidelně je zobrazuje na internetu. Základní informace o síti PlanetLab, její stručnou historii a několik nejdůležitějších projektů naleznete v první kapitole.

Druhá kapitola obsahuje základní informace o službách ping a SSH, jejichž dostupnost je testována již zmíněnou aplikací.

Kapitola třetí se pak zaobírá jazykem Bash a příkazovým řádkem operačního systému Linux, ve kterém je zpracován výsledný skript. Tedy skript, který každou hodinu testuje servery v síti PlanetLab a informace o jejich dostupnosti aktualizuje v textovém souboru.

Ve čtvrté kapitole jsou popsány principy a druhy paralelizace procesů, díky které se podařilo optimalizovat vytvořený skript a snížit tak pracovní dobu této aplikace na minimum.

Pátá kapitola již detailně popisuje práci na uvedeném skriptu, další možnosti využití tohoto skriptu, jeho formu a výstup.

V následující šesté kapitole naleznete výsledky měření, které je realizováno právě zmíněným skriptem.

Poslední sedmá kapitola obsahuje stručné shrnutí a zhodnocení celého projektu.

1 EXPERIMENTÁLNÍ SÍŤ PLANETLAB

PlanetLab je celosvětová počítačová síť sloužící pro výzkum a ověřování nových typů síťových aplikací. Její servery jsou rozloženy po celém světě, čímž umožňují testování v reálném prostředí celosvětového Internetu. Přístup do sítě PlanetLab mohou získat pouze instituce, které se do projektu samy zapojí, většinou jde o univerzity a vědecká pracoviště, případně vývojová centra velkých firem. Každý uživatel má pak možnost vytvořit si ze zařízení PlanetLabu vlastní síť, na které bude testovat svůj určitý projekt. Cílem PlanetLabu je přispět k řešení závažných problémů, jako je například nespolehlivá stabilita přenosových cest, stále častější pirátské útoky na servery nebo nedostatečný adresový prostor [1].

1.1 Terminologie

Node: je server, na kterém pracují služby sítě PlanetLab. Každý takový uzel musí mít alespoň jednu nesdílenou IP adresu.

Site: je fyzické místo, kde se nachází jeden nebo více uzlů (nodes) sítě PlanetLab. Typicky se jedná o univerzitu nebo sídlo firmy.

Slice: je účet unixového terminálu s přístupem na přednastavené uzly (nodes) sítě PlanetLab. Přístup k jednotlivým uzlům lze upravit na webové stránce sítě.

User: Uživatelem sítě PlanetLab je každý, kdo vyvíjí a testuje aplikace v této síti.

1.2 Historie sítě PlanetLab

V březnu 2002 Larry Peterson z Princeton University a David Culler z Intel Research a University of California at Berkeley zorganizovali setkání výzkumných pracovníků se zájmem o globální síťové služby a navrhli PlanetLab jako komunitní testovací platformu. Tohoto setkání se zúčastnilo 30 výzkumníků z Massachusetts Institute of Technology, Washington University, Rice, Berkeley, Princeton, Columbia, Duke, CMU (Carnegie Mellon University) a z Utahu. David Tennenhouse z Intel Research souhlasil s úvodním financováním projektu se 100 servery. V říjnu 2002 je již v provozu všech 100 uzlů na 42 místech a je tedy hotov úvodní návrh rozestavení uzlů. Poté již rozvoj sítě PlanetLab nabral rychlý spád. V září 2003 už PlanetLab dosáhl hranice 200 uzlů, a už v prosinci přesáhl počet uzlů 300. V lednu 2004 bylo formálně vytvořené PlanetLab konsorcium, neboli skupina akademických, průmyslových a vládních organizací spolupracujících na podpoře a růstu sítě PlanetLab. Členy PlanetLab se stalo i mnoho dalších univerzit, významná výzkumná

pracoviště firem z oblasti informačních technologií (jako jsou HP, Intel, France Telecom), organizace zajišťující provoz Internetu v akademických komunitách (jako jsou Internet2-USA, Canarie-Kanada, Cernet-Čína) a další národní pracoviště se širokým výzkumným posláním (INRIA-Francie, GIST-Korea) atd. Dnes (květen 2016) se PlanetLab skládá ze 1353 uzlů v 717 organizacích [1].

1.3 Projekty v síti PlanetLab

Hlavním cílem sítě PlanetLab je sloužit jako testovací prostředí, které je schopné simulovat reálné podmínky celosvětového Internetu. Díky tomu zde mohou vývojářské skupiny experimentovat s nejrůznějšími síťovými službami jako je například sdílení dat, vrstvení QoS (Quality of Service) nebo detekce anomálií v síti a to vše přímo v reálných podmínkách. To ze sítě PlanetLab dělá naprosto unikátní testovací prostředí, ve kterém o příliv projektů není nouze [2].

Následující příklady ukazují dimenze veličin, se kterými projekty kalkulují. Jde o několik největších projektů v síti PlanetLab:

- Projekt OceanStore - Jedná se o jeden z největších projektů sítě PlanetLab, který řeší celosvětovou paměť. Uživatelé této paměti se pak nestarají o to, kde jsou jejich data uložena [3].
- CoBlitz, CoWeb a CoDeploy - Služby, které umožňují paralelní přenos velkých objemů dat skrz distribuovanou síť, což je využitelné např. pro distribuci zdrojů operačních systémů.
- CoDNS - Služba, která se v síti stará o DNS (Domain Name System). Tento DNS je pak odolnější proti útokům a nespolehlivosti serverů[4].
- Projekt DHARMA - Již celý název této zkratky (Distributed Home Agent for Remote Mobile Access) napovídá, že se v tomto projektu jedná o mobilní připojení. Přesněji tento projekt řeší situace, kdy je třeba zajistit přechod z různých sítí na mobilní připojení bez nutnosti opětovně navazovat spojení [5].
- Distribuční systém Coral - Tento projekt je určen pro efektivní distribuci dat. Využívají jej uživatelé, kteří mají pomalé připojení k síti, ale přesto potřebují distribuovat své informace vystavené na jejich webových serverech širokému okruhu dalších uživatelů [6].

Tím ale výčet projektů zdaleka nekončí. Dále bylo v PlanetLabu testováno prostředí pro projekt IrisNet (Internet-scale Resource Intensive Sensor Network Services), což je síť osobních webových kamer, u které se předpokládalo, že by v ní mohlo být připojeno až 300 milionů lidí. Co se týče skromnějších projektů, tak se obvykle uvažuje jen o metropolitních dimenzích s cca 1,5 milionem lidí. Nicméně

i metropolitní rozměr znamená testování mnoha typů serverů. Takové servery by například mohly hlásit stav obsazenosti parkovišť nebo stav provozu na křižovatkách a podobně. V takovém případě by pouze uchovávání informací o lokalizaci objektů vyžadovalo asi 25 000 aktualizací za sekundu [7].

Mnoho z těchto i dalších projektů je přímo svázáno, případně navazuje na další výzkumné projekty, které patří do třídy DHT (Distributed Hash Table). Což jsou metody, které spadají do jedné z nejvíce se rozvíjejících oblastí základního výzkumu s velkou publikační aktivitou [7].

2 SLUŽBY PING A SSH

V následujícím textu jsou představeny služby ping a SSH. Dostupnost těchto služeb jsem měl za úkol testovat na zařízeních sítě PlanetLab v praktické části této práce.

2.1 Služba ping

Ping (Packet InterNet Groper) je ve své podstatě program, který periodicky odesílá IP (Internet Protocol) datagramy z jednoho zařízení připojeného do sítě a očekává odezvu od druhého zařízení, na který byl tento datagram vyslán. Při úspěšném obdržení odpovědi od druhého zařízení vypíše délku zpoždění každého vyslaného datagramu a na závěr statický souhrn všech přijatých nebo nepřijatých odpovědí. Pokud odpověď od druhého zařízení zpět k prvnímu nedoputuje, je možné buď, že zařízení mezi sebou nemají spojení nebo že je služba ping na druhém zařízení zakázána, což ovšem není příliš časté. Ping tedy umožňuje jednoduchou analýzu funkčnosti spojení, mezi dvěma síťovými zařízeními, které komunikují prostřednictvím rodiny protokolů TCP/IP (Transmission Control Protocol/Internet Protocol).

Program ping využívá protokol ICMP (Internet Control Message Protocol), který nezaručuje doručení zprávy. Parametrem tohoto programu může být IP adresa nebo případně DNS jméno síťového rozhraní, u kterého chceme prověřit jeho dostupnost. Je-li jako parametr zadáno právě DNS jméno, pak je nejprve přeloženo na IP adresu pomocí dostupného DNS serveru. ICMP zprávy jsou odesílány na IP adresu cílového rozhraní, které chceme otestovat, a v určitém limitu, typicky trvajícím 3 sekundy, se očekává odpověď. Pokud odpověď není přijata během tohoto limitu, je testované rozhraní označeno za nedostupné. Typicky je vypsána hláška „Vypršel časový limit žádosti“. Každá taková ICMP zpráva je očíslována, díky čemuž je možné jednoduše identifikovat jednotlivé odpovědi nebo jejich ztrátu. Program průběžně vypisuje výsledky odpovědí vzdáleného zařízení spolu se zpožděním [8].

2.2 Služba SSH

Roku 1995 navrhl finský student Tatu Ylönen první verzi protokolu SSH-1 jako reakci na útok odchytáváním hesel na počítačovou síť helsinské Technické univerzity (Helsinki University of Technology, Finsko). Následně pak tuto první verzi SSH-1 uveřejnil včetně zdrojových kódů jako freeware. Již na konci roku 1995 používalo tuto první verzi SSH okolo 20 000 uživatelů v padesáti zemích.

To Ylönena dovedlo k založení společnosti SSH Communications Security právě v prosinci roku 1995 a k začátku vyvíjení nového SSH a dalších bezpečnostních

nástrojů. Původní verze SSH používala části jiných freeware softwarů. Další verze se již odprostily od těchto závislostí a začaly se vydávat pouze pod uzavřenou licencí, což později vedlo k vývoji nového SSH jinými vývojáři. Snahou bylo poskytnout SSH zase jako freeware.

V následujícím roce byla vytvořena nová verze protokolu SSH-2, která je však nekompatibilní s SSH-1. V této nové verzi byla zvýšena bezpečnost například díky přísné kontrole integrity dat pomocí MAC funkce nebo díky výměně klíčů pomocí Diffie-Hellman algoritmu (anglicky Diffie-Hellman key exchange). Nově u protokolu SSH-2 je také možnost řídit libovolný počet shellů pomocí jednoho SSH spojení [11].

Jak již bylo zmíněno, někteří vývojáři nebyli ochotni platit za SSH-2 a tak se v roce 1999 vrátili ke starší verzi původního SSH (1.2.12), která byla jako poslední vydána jako open source software. Na základě tohoto staršího SSH se pak několik vývojářů pokoušelo vytvořit nový volně šiřitelný SSH, který by se mohl rovnat SSH-2. Výsledkem toho byl OSSH, který vyvinul Švéd Björn Grönvall. Následně vývojáři OpenBSD rozvinuli tuto verzi OSSH, udělali v ní rozsáhlé změny a vytvořili OpenSSH, který umožňoval portování i na jiné operační systémy.

V roce 2005 se stalo OpenSSH jednou z nejoblíbenějších verzí SSH a OSSH se stalo zastaralým. V následujícím roce 2006 byla verze SSH-2 navržena jako Internetový standard [12].

Tímto bych rád ukončil zmínku o historii SSH a chtěl bych představit, co SSH vlastně znamená a k čemu se používá. Zkratkou SSH (Secure Shell) se v informatice označuje program společně se zabezpečeným komunikačním protokolem, který se používá v počítačových sítích k zabezpečení dat při přenosu přes nebezpečnou síť, jako je například Internet. Díky tomu umožňuje bezpečnou komunikaci mezi dvěma vzdálenými počítači, která se většinou využívá pro zprostředkování přístupu k příkazovému řádku, kopírování souborů pomocí protokolu SCP (Secure Copy) a též jakýkoliv obecný přenos dat s využitím síťového tunelování. SSH se však nejčastěji používá ke správě vzdáleného zařízení, právě prostřednictvím přístupu k jeho příkazovému řádku.

SSH byl navržen z důvodu stále vzrůstajícího počtu síťových zařízení a tím i vzrůstající hrozby nejrůznějších útoků na nezabezpečenou síťovou komunikaci, kterou zprostředkovával například telnet a další nezabezpečené shelly. Tyto shelly při autentizaci uživatele posílají skrz počítačovou síť nezabezpečené heslo a umožňují tak jeho jednoduché odposlechnutí. Tímto způsobem pak může útočník naprosto převzít kontrolu nad vzdáleným zařízením.

K autentizaci uživatele u SSH se používá zabezpečené heslo, klíč RSA (iniciály autorů Rivest, Shamir, Adleman, RSA je šifra s veřejným klíčem), metoda keyboard-interactive (server pošle klientovi jednu nebo více výzev a klient na ně odpoví pomocí klávesnice) a metoda GSSAPI (poskytuje rozšiřitelné rozhraní pro autentizaci po-

mocí externích mechanizmů). Tyto metody jsou většinou kombinovány mezi sebou. Například k zařízením v síti PlanetLab je možné se připojit pouze pokud znáte heslo a máte již ověřený veřejný klíč. SSH tak zajišťuje autentizaci obou účastníků komunikace [9].

Využití SSH je opravdu široké. Jak již bylo zmíněno lze jej použít na administraci serverů, avšak dále na založení VPN (virtuální privátní sítě), přesměrování portů TCP nebo brouzdání po webu přes proxy se šifrováním. Díky síťovému tunelování je tento komunikační kanál pro tyto účely velmi bezpečný.

U protokolu SSH server standardně naslouchá na portu TCP/22 [10].

3 PROGRAMOVÉ PROSTŘEDÍ

V této kapitole je představeno prostředí, ve kterém byla vytvořena aplikace pro pravidelné monitorování serverů v Internetu. Jedná se zejména o základní informace o Bashi a o linuxovém příkazovém řádku.

Bash (akronym pro Bourne again shell) je jeden z unixových shellů, který je ovládán příkazovým řádkem. Základ syntaxe jazyka Bash je převzat z původního Bourne shellu (sh), díky čemuž lze převážnou většinu sh skriptů spustit v Bashi bez modifikace kódu. Existuje ale pár výjimek, které vznikají odlišnou implementací systémových příkazů nebo odlišnou interpretací méně používaných příkazů. Bash dále převzal nápady i z Korn shellu (ksh) a C shellu(csh). Jedním z nejvýznamnějších je například historie vložených příkazů, kterou lze procházet šipkami nahoru a dolů nebo editace jednotlivých řádků. Dalšími převzatými nápady jsou proměnné $RANDOM a POSIXové (standart pro unixové systémy) dynamické vkládání příkazů pomocí syntaxe $(...). Možná právě proto se někdy název Bash označuje jako zkratka pro stlučení (anglicky bashing) všech výhod výše uvedených shellů. Další výhodou Bashe, která výrazně usnadní práci, je automatické doplňování rozepsaného příkazu, cesty k souboru nebo proměnné, případně ukázání všech možností doplnění. Tuto funkci ovládá uživatel pomocí klávesy tabulátor [13].

Jak již bylo řečeno, Bash získal mnoho vlastností z Bourne shellu, avšak navíc má mnoho rozšíření i v syntaxi příkazů. Jedním z těchto rozšíření je možnost provádět celočíselné operace bez volání externích procesů. Pokud tedy vložíte celočíselný příklad do dvojitých závorek a před tento výraz vložíte značku dolaru, Bash tento příklad dokáže spočítat a vypsat výsledek do vámi určené proměnné, souboru nebo ho vypsat přímo do konzole. Další výhodou oproti tradičnímu Bourn shellu je přesměrování standardního výstupu a standardního chybového výstupu operátorem &> [13].

Bash je spustitelný na většině unixových operačních systémů. Je však možné jej spustit i v systému Microsoft Windows s použitím subsystému pro unixové aplikace nebo pomocí speciálního softwaru. V operačním systému Linux práci v Bashi zprostředkovává příkazový řádek pojmenovaný jako terminál. V podstatě veškerá práce na vytvoření aplikace pro monitorování serverů v Internetu probíhala v tomto příkazovém řádku.

Příkazový řádek (zkratka CLI, anglicky Command Line Interface) je ve své podstatě uživatelské rozhraní, jehož prostřednictvím uživatel komunikuje s operačním systémem nebo s programy tak, že vkládá příkazy vypsáním celého jejich názvu a stiskem klávesy „Enter“. Ihned po spuštění příkazového řádku je zobrazen název standardního adresáře, ve kterém se uživatel pohybuje a blikající kurzor. Po vložení příslušného příkazu je spuštěn odpovídající program, který je pak schopen vypiso-

vat do konzole informace o svém běhu, výsledky operací nebo si může vyžádat další vstup a podobně. Po dokončení přikázané operace je ihned znovu zobrazen blikající kurzor značící připravenost příkazového řádku na další instrukce.

Pro představu se příkazový řádek nedokáže přizpůsobit celé ploše obrazovky a k ovládání nelze využívat myš, tak jako je to možné například v grafickém uživatelském prostředí. Příkazový řádek lze tedy ovládat pouze klávesnicí a výpisem přesně definovaných příkazů.

4 PARALELIZACE PROCESŮ

Paralelizace je slovem mnoha významů, nicméně my se budeme bavit o paralelizaci procesů ve smyslu optimalizace počítačového programu. Tato zmíněná optimalizace spočívá v rozdělení jednoho velkého výpočetního problému na několik menších problémů, které jsou řešeny z pohledu uživatele současně. Z pohledu počítače to však není úplně pravda, protože na jednom jádře procesoru vždy běží pouze jeden proces a ostatní čekají, dokud není proces dokončen nebo přerušen. Avšak i proces, který je v běhu, může čekat na výsledky zadané instrukce, což kupříkladu u testu SSH, kdy můžeme na navázání spojení čekat až několik sekund, není žádoucí. Obzvláště, pokud navíc následující proces čeká na výsledky předchozího nebo pokud je zkrátka spouštěn sekvenčně až po dokončení procesu předchozího, tak jak je tomu standardně právě u skriptů a práce s příkazovým řádkem. Pokud jsou pak takových procesů stovky, sekundy se nám nepříjemně a zbytečně naskládají až na hodiny práce programu. Proto je vhodné v průběhu čekání na připojení spustit další proces, což značně zrychlí chod programu.

Další, spíše hardwarové, prvky sloužící k paralelnímu zpracování výpočtu mohou být například počítač s více procesory, několik počítačů v síti, specializovaný hardware nebo kombinace těchto prvků. My se však dále zaměříme na softwarové způsoby paralelizace a jejich rozdělení.

4.1 Typy paralelizace

Pravděpodobně nejznámější klasifikace paralelních systémů je tzv. Flynnova klasifikace, která vznikla v roce 1966. Tato klasifikace spočívá v rozdělení systémů na základní 4 typy.

- SISD (Single instruction, single data) - Jak již význam zkratky napovídá, jedná se o typ zpracovávání dat sekvenčně podle jednoho programu. Jinými slovy se vykonává jedna instrukce programu po druhé. Jde o typ bez jakékoli paralelizace.
- SIMD (Single instruction, multiple data) - V tomto případě je zpracováváno více dat jedním programem. Typicky jde o program používající několik jader procesoru, tudíž všechny procesory současně vykonávají stejnou instrukci, ale zpracovávají různá data. Tento typ je použit v praktické části práce.
- MISD (Multiple instruction, single data) - Jde o typ prakticky nepoužívaný, který vznikl v podstatě jen z důvodu úplnosti klasifikace. Teoreticky by šlo o více programů, která zpracovávají pouze jedna data několikrát.
- MIMD (Multiple instruction, multiple data) - Typ kdy několik programů zpracovává několik různých dat. Typicky jde o vícejádrový systém, kde na každém

jádře pracuje jiný program a vykonává odlišné instrukce s odlišnými daty [14].

Aplikace lze také klasifikovat podle toho, jak často jejich dílčí úlohy komunikují mezi sebou. Tak lze paralelizaci dělit dále jako jemnozrnnou, kdy je třeba, aby spolu úlohy komunikovaly několikrát za sekundu, hrubozrnnou, kdy spolu úlohy nepotřebují komunikovat několikrát za sekundu, ale komunikují spolu méně často a poslední možnost je jednoduchá paralelizace. V tomto případě spolu úlohy nekomunikují buď vůbec a nebo velmi vzácně. Jednoduché paralelizace je využito v praktické části práce. Jedná se o nejjednodušší možnost paralelizace.

4.2 Postup při návrhu paralelizace

Jak již bylo zmíněno, paralelizaci programu je nutno vnímat jako optimalizaci. Proto je nutné při návrhu programu nejprve vyvinout nejjednodušší sekvenční algoritmus bez optimalizace, která by mohla zcela zbytečně poničit čistý návrh algoritmu. Navíc, pokud nemáme základní sekvenční kód, nelze poté proměřit přínos paralelizace oproti optimalizovanému kódu.

Pokud máme základní funkční kód, přistupujeme k optimalizacím pouze pokud základní algoritmus nesplňuje naše požadavky. Je vhodné nejdříve optimalizovat jen sekvenční kód a k paralelizaci přikročit pokud optimalizace sekvenčního kódu nepostačuje. Také musíme zjistit, zdali je pro nás paralelizace vůbec výhodná, u některých typů algoritmů paralelizace pouze udělá kód programu složitějším a přitom není schopna běh programu urychlit. K tomu může dojít, pokud program pracuje na pomalém hardwaru nebo pokud následující úloha pracuje s výsledky předchozí úlohy a tudíž takové úlohy nemohou být spuštěny zároveň. Pokud je ovšem paralelizace vhodná a my se pro ni rozhodneme, je nutné během její implementace provádět průběžné testy a kontrolovat tím, zda optimalizovaný kód nevytváří odlišné výsledky. Nakonec je vhodné proměřit výslednou efektivitu paralelizace.

Lze také postupovat podle Iana Fostera, který navrhl čtyři kroky vedoucí k návrhu paralelního algoritmu. Tento postup je naprosto nejběžnější při návrhu paralelních algoritmů. Prvním krokem je partitioning, tedy rozdělení úlohy algoritmu na několik malých úloh, které by bylo možné spouštět současně. Druhým krokem je communication, neboli komunikace. V tomto kroku je třeba stanovit, jak spolu budou nebo nebudou jednotlivé podúlohy komunikovat. Ve třetím kroku zvaném aglomeration je pak nutné sloučit malé podúlohy do větších, aby došlo k redukci komunikace mezi podúlohami a tak k redukci počtu potřebných komunikačních kanálů. Získáme tak úlohy zvané tasks. Platí, že čím podobnější si tyto úlohy jsou, tím je paralelizace efektivnější. Posledním čtvrtým krokem zvaným mapping je přidělování jednotlivých tasks příslušným procesorům nebo výpočetním jednotkám [15].

5 VYTVOŘENÝ PROGRAM

Kapitola 5 popisuje vlastní postup při tvorbě aplikace, která slouží k monitorování serverů v Internetu. Tato aplikace byla vytvořena v prostředí Linux ve formě Bash skriptu. Úkolem aplikace je zjistit, která zařízení v síti PlanetLab odpovídají na ping a SSH připojení, a která zařízení na tyto služby neodpovídají.

5.1 Postup tvorby skriptu a jeho funkce

Nejdříve je nutné vygenerovat si veřejný a soukromý klíč, aby bylo možné připojit se přes SSH do sítě PlanetLab. Tento krok je nutný udělat v první řadě, protože síti PlanetLab trvá 24 hodin, než se nový veřejný klíč rozšíří na všechny její uzly. Vygenerování klíče je možné přímo přes terminál v operačním systému Linux nebo v programu Puttygen, pokud používáte operační systém Windows. Další postup je již uváděn pro operační systém Linux, konkrétně byla aplikace vytvořena v systému Ubuntu. Funkci programu popisuje vývojový diagram, který je zobrazen na Obr. 5.1.

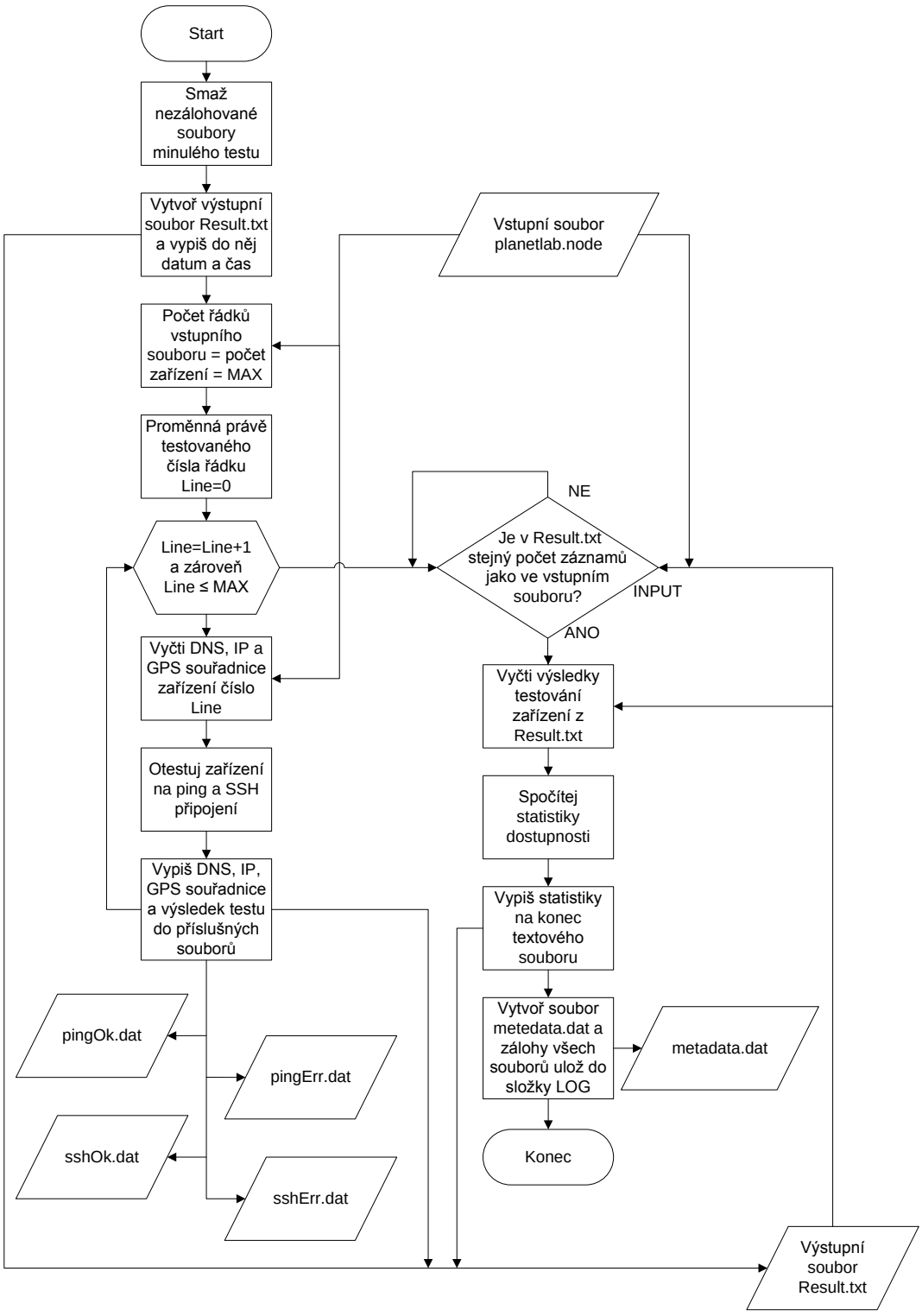
Skript, tedy aplikaci, která byla objektem zadání této práce, je možné psát v jakémkoli textovém editoru, protože jde v podstatě o textový soubor. Je možno využít poznámkový blok nebo například PSPad editor, který je schopen zvýrazňovat syntaxi jazyka Bash. K vytvoření přiložené aplikace byl využit program gedit, který také dokáže zvýrazňovat syntaxi. Zároveň je tento program nainstalován již na základní verzi systému Ubuntu.

Uvedený skript začíná řádkem *#!/bin/bash*, což je řádek, který odkazuje na cestu do adresáře bin/bash a říká tak systému, kde najde program ke spuštění daného skriptu. Na dalších třech řádcích jsou zapsány do proměnných názvy vstupního a výstupního souboru (input, output) a umístění složky sloužící pro uchovávání předešlých měření. Toto zapsání do proměnných je provedeno z důvodu snazší práce se skriptem nebo jeho snazší úpravy.

Následuje příkaz pro smazání předchozího výstupního souboru a dalších 5 příkazů pro smazání dalších předchozích datových souborů. Díky těmto příkazům dojde v podstatě k přepsání výstupního a datových souborů při novém testování. Předešlé výsledky však neztratíme, ty se zkopírují do složky LOG, ale o tom později.

Poté si aplikace spočítá kolik bude testovat zařízení prostřednictvím uložení DNS názvů ze vstupního souboru, který obsahuje seznam zařízení, do proměnné. Na tento úkon použije programy sed a awk. Následně pomocí příkazu wc –l spočítá počet řádků v proměnné, tedy počet DNS jmen.

Dále je vytvořen textový soubor a na jeho první řádek je zapsáno datum a čas



Obr. 5.1: Vývojový diagram aplikace pro monitorování serverů v Internetu.

hned za znak „#“. Křížek je vepsán z důvodu snadného odlišení času od dat textového souboru. Na druhý řádek souboru je vypsána jednoduchá vysvětlivka formátu výstupu. Na dalším řádku skriptu je uloženo datum a čas začátku testování do proměnné.

Následuje hlavní tělo aplikace. Jde o metodu, která testuje zda-li testované zařízení odpovídá na ping a SSH a výsledek zapisuje do textového souboru v podobě 0, pokud služba odpovídá nebo 1, pokud služba neodpovídá. Na začátku této metody je vepsán kód s příkazem *nslookup*. Tento kód ověřuje, že existuje zadané DNS jméno. Vyhneme se tím chybovým hláškám při testu zařízení na ping. Následně je již postupně testován ping na zařízení a úspěšnost SSH připojení. Uvedená metoda je zobrazená ve výpisu 5.1. Dále je nutné popsat nastavení SSH, která jsou součástí zmíněné metody. Všechna nastavení jsou nutná pro správnou funkci testu SSH připojení. Nastavení *PrefferedAuthentications=publickey* říká vzdálenému zařízení, že při připojování prostřednictvím SSH se má použít ověřování veřejným klíčem, *PasswordAuthentication=no* říká, aby zařízení nežádalo heslo neboli tzv. passphrase, položka *ConnectTimeout=10* nastavuje čas, který se aplikace pokouší připojit přes SSH na vzdálené zařízení na 10 sekund, což je více než dostačující doba. Vypozoroval jsem, že pokud se SSH připojení nesestaví do 10 sekund, tak už se nesestaví vůbec. Navíc zbytečně více nastavených sekund dramaticky zpomaluje chod programu. Následující položka *StrictHostKeyChecking=no* zakazuje dotaz serveru na kontrolní úhoz uživatele do klávesnice, položka *UserKnownHostsFile=/dev/null* pak přikazuje, aby si server neukládal žádné informace o pokusu připojení. Dále již uvedený SSH příkaz obsahuje pouze údaje k připojení a cestu k adresáři, ve kterém je uložený soukromý klíč. Po testu dostupnosti stanice službou ping a dostupnosti SSH připojení, následuje příkaz, který připíše DNS jméno, IP adresu, GPS (Global Positioning System) souřadnice a výsledek testování určitého zařízení na konec textového souboru.

Na konci testu určitého zařízení se vytvoří další dva výstupní soubory, pokud již nejsou vytvořeny, do kterých se připíše DNS jméno právě testované stanice. Celkem jsou tohoto druhu čtyři soubory, pro každý stav jeden soubor. Tudíž pokud máme zařízení, které je dostupné na ping a na SSH je nedostupné, zapíše se DNS jméno této stanice do souborů pingOk.dat a sshErr.dat. Pokud je zařízení na ping nedostupné a na SSH dostupné, pak se zapíše do souborů pingErr.dat a sshOk.dat atp. Tímto postupem tedy na konci testování získáme čtyři soubory, které obsahují všechna DNS jména jednotlivých skupin zařízení rozdělených podle jejich dostupnosti na služby. Blíže jsou tyto soubory popsány v kapitole 5.2.

```
1  monitoring() #metoda pro ping a ssh {
2  echo Testing $1
3  unknowndomain=0
4  nslookup $1 > /dev/null
5  unknowndomain=$?
6  if [ $unknowndomain -eq 0 ];then
7    ping -c 1 $1 > /dev/null
8    if [ $? -eq 0 ]; then
9      ssh -q -o "PreferredAuthentications=publickey" -o "
          PasswordAuthentication=no" -o "ConnectTimeout=30" -o "
          UserKnownHostsFile=/dev/null" -o "StrictHostKeyChecking=no"
          -i ~/xpospi81/.ssh/id_rsa cesnet_feec@$1 exit
10     if [ $? -eq 0 ]; then
11      echo -e "$1\t$ip\tping\t0\tssh\t0\t$latitude\t$longitude" >>
           $output
12      echo $1 >> pingOk.dat
13      echo $1 >> sshOk.dat
14     else
15      echo -e "$1\t$ip\tping\t0\tssh\t1\t$latitude\t$longitude" >>
           $output
16      echo $1 >> pingOk.dat
17      echo $1 >> sshErr.dat
18     fi
19   else
20     ssh -q -o "PreferredAuthentications=publickey" -o "
          PasswordAuthentication=no" -o "ConnectTimeout=30" -o "
          UserKnownHostsFile=/dev/null" -o "StrictHostKeyChecking=no"
          -i ~/xpospi81/.ssh/id_rsa cesnet_feec@$1 exit
21     if [ $? -eq 0 ]; then
22      echo -e "$1\t$ip\tping\t1\tssh\t0\t$latitude\t$longitude" >>
           $output
23      echo $1 >> pingErr.dat
24      echo $1 >> sshOk.dat
25     else echo -e "$1\t$ip\tping\t1\tssh\t1\t$latitude\t$longitude"
          >> $output
26      echo $1 >> pingErr.dat
27      echo $1 >> sshErr.dat
28     fi
29   fi
30 else echo -e "$1\t$ip\tping\t1\tssh\t1\t$latitude\t$longitude" >>
     $output
31     echo $1 >> pingErr.dat
32     echo $1 >> sshErr.dat
33 fi}
```

Výpis kódu 5.1: Metoda Monitoring.

Poté je nutné spustit výše uvedenou metodu, což je provedeno následujícím cyklem viz. výpis 5.2. V tomto cyklu jsou nejdříve načteny všechny potřebné údaje o určitém zařízení. Tyto údaje se poté ukládají do proměnných, s nimiž následně pracuje metoda Monitoring. Metoda Monitoring je spuštěna po uložení údajů do proměnných s parametrem DNS jména. Na řádku spouštění metody je možné si všimnout také příkazu disown, který zajistí, že cyklus nebude čekat na dokončení testu předchozího zařízení, ale spouští paralelně test následujícího zařízení. Tím dojde k výraznému ušetření času při testování velkého množství uzlů. Poslední příkaz v cyklu provádí uspání programu na 100 ms po spuštění testu každého zařízení. Tento příkaz zabrání téměř současnému spuštění testu všech zařízení PlanetLabu a tím pádem i obrovské zátěži sítě a hardwaru počítače. Tento problém je dále rozveden v kapitole 5.4.

```
1  echo Testing $max nodes, please "wait".
2  for((line=1; line<=$max; line++))
3  do
4    dns=$(sed '/#/d' $input| awk -F '\t' '{print $3}' | sed -n ''"
       $line"'p')
5    ip=$(sed '/#/d' $input| awk -F '\t' '{print $2}' | sed -n ''"
       $line"'p')
6    latitude=$(sed '/#/d' $input| awk -F '\t' '{print $10}' | sed -n
       ''"$line"'p')
7    longitude=$(sed '/#/d' $input| awk -F '\t' '{print $11}' | sed -n
        ''"$line"'p')
8    monitoring $dns & disown
9    sleep 0.1
10 done
11 pom=0
12 until [ $pom -eq $max ]; do
13   dnsc=$(sed '/#/d' $output|awk -F '\t' '{print $1}')
14   pom=$(echo "$dnsc"|wc -l)
15 done
```

Výpis kódu 5.2: Cyklus spouštějící metodu Monitoring.

Po proběhnutí testu všech zařízení je spuštěn cyklus, který zjišťuje, zda jsou údaje o všech zařízeních zapsány ve výstupním textovém souboru *Result.txt*. Pokud je tato podmínka splněna, začnou se počítat statistiky z právě vytvořeného souboru *Result.txt*, které jsou následně dopsány na konec tohoto souboru.

K výpočtu statistik slouží část skriptu označená jako Statistics, kterou si můžete prohlédnout ve výpisu 5.3. Tato část kódu v podstatě prochází nový výstupní soubor řádek po řádku a přičítá do patřičných proměnných zařízení po zařízení podle toho

jaký je status u ping a SSH (0 a 1) v daném řádku. V závěru tedy dostaneme proměnné, ve kterých jsou uloženy počty zařízení rozdělených podle jejich dostupnosti služeb.

```
1  echo "Statistics in progress"
2  ### STATISTICS ###
3  dnsc=$(sed '/#/d' $output|awk -F '\t' '{print $1}')
4  max=$(echo "$dnsc"|wc -l)
5  up=0
6  down=0
7  pingok=0
8  sshok=0
9  for((line=1; line<=$max; line++))
10 do
11 pingpom=$(sed '/#/d' $output| awk -F '\t' '{print $4}' | sed -n ''"
       $line"'p')
12 sshpom=$(sed '/#/d' $output| awk -F '\t' '{print $6}' | sed -n ''"
       $line"'p')
13 if [ $pingpom -eq 0 ];then
14   if [ $sshpom -eq 0 ];then
15     up=$((up+1))
16   else pingok=$((pingok+1))
17   fi
18 else
19   if [ $sshpom -eq 0 ];then
20     sshok=$((sshok+1))
21   else down=$((down+1))
22   fi
23 fi
24 done
25 echo "# up-$up nodes | only ping ok-$pingok nodes | only ssh ok-
       $sshok nodes | down-$down nodes | total-$max nodes" >> $output
```

Výpis kódu 5.3: Výpočet statistik.

Po výpočtu statistik následuje část skriptu označená jako Metadata. V této části skriptu je vytvořen soubor *metadata.dat*. Dále jsou v této sekci vytvořeny zálohy všech šesti výstupních souborů tak, že jsou nakopírovány do složky LOG, která leží ve stejném umístění jako testovací skript. Aby byly tyto soubory časově snadno rozlišitelné, je každý soubor při kopírovaní do složky přejmenován na stávající jméno souboru obohacené o datum a čas proběhlého testu. Pro představu tedy ve složce nemáme soubor *pingOk.dat* ale soubor *„pingOk_datum_čas.dat“*. U hlavního výstupního souboru *Result.txt* je však postup odlišný. Tento soubor je již vytvořen už na začátku skriptu s názvem obsahujícím datum a čas a až na úplném konci

programu je vytvořen soubor *Result.txt*, který zůstává v domovském umístění a původní soubor s datem a časem je přesunut do složky LOG. Toto řešení je zdůvodněno v kapitole 5.4.

Po vytvoření záloh je celý program u konce a je tedy do konzole vypsáno ukončení skriptu s celkový počtem otestovaných stanic.

5.2 Výstup skriptu.

Výstupem dříve zmíněné aplikace pro monitorování serverů v Internetu je textový soubor, obsahující všechna potřebná data pro navazující práci „Výzkumná síť PlanetLab“ Matěje Grejtáka. Jedná se tedy o DNS jména, IP adresy, GPS souřadnice a zejména stav dostupnosti stanic a jejich vzdáleného SSH připojení. S těmito daty dále Matej Grejták pracuje na vytvoření mapy, která zobrazuje polohu a stav uzlů sítě PlanetLab po celém Světě. Část již zmíněného textového souboru je zobrazena ve výpisu 5.4.

```
1  #Thu Apr 21 11:00:01 UTC 2016
2  #File format: 1.column-DNS-device name; 2.column-IP address; 4.
       column: 0-ping succesful, 1-ping unavailable; 6.column: 0-ssh
       succesful atempt, 1-ssh unavailable; 7.column-GPS latitude; 8.
       column GPS longitude.
3  planetlab1.poznan.rd.tp.pl  195.116.60.113  ping  1 ssh 1 52.33 17
4  planetlab3.poznan.rd.tp.pl  195.116.60.115  ping  1 ssh 1 52.33 17
5  planetlab1.unl.edu  129.93.229.138  ping  0 ssh 0 40.8  -96.6667
6  planetlab1.cnu.ac.kr  168.188.48.112  ping  1 ssh 1 37  127.5
7  planetlab2.esprit-tn.com  41.225.7.4  ping  0 ssh 1 36  10
8  planetlab4.poznan.rd.tp.pl  195.116.60.116  ping  1 ssh 1 52.33 17
9  datacomngn.cnu.ac.kr  168.188.48.26 ping  1 ssh 1 37  127.5
10 planetlab-2.unk.edu 144.216.2.53  ping  1 ssh 1 40.7  -99.1
11 planetlab2.utt.fr 194.254.215.12  ping  0 ssh 0 48.2693 4.06739
12 planetlab-1.unk.edu 144.216.2.52  ping  1 ssh 1 40.7  -99.1
13 planet-2.utn.edu.ar 170.210.22.10 ping  1 ssh 1 -34.58  -58.48
14 planetlab2.tsuniv.edu 206.23.240.29 ping  1 ssh 1 36.1659 -86.8313
15 # up-167 nodes | only ping ok-191 nodes | only ssh ok-10 nodes |
       down-647 nodes | total-1015 nodes
```

Výpis kódu 5.4: Část výstupního textového souboru *Result.txt*.

Výstupní soubor se skládá z data a času na prvním řádku, pokračuje počtem řádků, který odpovídá počtu testovaných zařízení (ve výpisu 5.4 je počet řádků testovaných zařízení záměrně zkreslen) a je zakončen řádkem, ve kterém jsou vypsány statistiky celého měření.

Statistiky tedy uvádějí kolik zařízení je v provozu s oběma dostupnými službami, kolik zařízení odpovídá pouze na službu ping, kolik pouze na službu SSH, kolik zařízení neodpovídá ani na jednu službu a kolik zařízení bylo otestováno celkem. Výpis statistik je zobrazen na konci výstupního textového souboru *Result.txt*. Řádek data a času, řádek s legendou a řádek se statistikami (první, druhý a poslední řádek souboru *Result.txt*) je označen na začátku symbolem „#“, aby bylo možné snadno odlišit test zařízení od ostatních náležitostí.

Uvedená aplikace však vytváří dalších 5 výstupních souborů, které budou postupně popsány. První soubor s názvem *Metadata.dat* uvádí na prvním řádku praktickou vysvětlivku struktury tohoto souboru. Na druhém řádku lze vyčíst datum a čas počátku testování zařízení sítě PlanetLab, na třetím řádku je vypsán čas ukončení testování. Další tři řádky vypisují celkový počet testovaných zařízení, počet zařízení dostupných na službu ping a na posledním řádku je vypsán počet zařízení dostupných na službu SSH. Soubor *metadata.dat* je zobrazen ve výpisu 5.5.

```
1 #File format: 1.line - beginning of program; 2.line - ending of
     program; 3.line - quantity of all nodes; 4.line - quantity of
     ping Ok nodes; 5.line - quantity of ssh Ok nodes.
2 Fri Apr 22 11:00:02 UTC 2016
3 Fri Apr 22 11:02:35 UTC 2016
4 1015
5 368
6 184
```

Výpis kódu 5.5: Výstupní datový soubor *metadata.dat*.

Následující 4 soubory obsahují DNS jména zařízení, které odpovídají příslušné skupině. Pro upřesnění tedy soubor *pingOk.dat* obsahuje DNS jména všech zařízení dostupných na ping a *pingErr.dat* obsahuje DNS jména všech zařízení na ping nedostupných. Stejně je tomu tak i u služby SSH v souborech *sshOk.dat* a *sshErr.dat*.

5.3 Pravidelné spouštění skriptu

Jedním z požadavků na aplikaci bylo spouštění testování pravidelně jednou denně. Tento úkon byl vyřešen pomocí programu Cron, který je již součástí linuxových operačních systémů. Tento program dokáže spouštět jakoukoli spustitelnou aplikaci v intervale, který je zapsán svou specifickou formou do souboru crontab.

Nyní bych rád popsal, jak spustit aplikaci pomocí Cronu. Příkazem crontab s příznakem -e, jako editovat, se lze dostat do zapisovacího módu crontabu. Nyní

je třeba zadat časový interval, který se zapisuje formou pěti časových údajů za sebou. První časový údaj značí minutu, druhý hodinu, třetí den v měsíci, čtvrtý měsíc a pátý den v týdnu. Všechny časové údaje se zadávají ve formě číslice, avšak měsíce a dny v týdnu je možné zadat i zapsáním prvních tří písmen z anglického názvu (např Jan-Dec nebo Mon-Sun). Pokud chceme uvést za daný časový údaj všechny možnosti, například chceme spouštět aplikaci každý měsíc, je praktické uvést místo všech hodnot jednoduše hvězdičku. Tudíž pokud chceme spouštět aplikaci každý den v poledne zadáme údaj jako "0 12 * * *" a následně umístění spustitelné aplikace, případně nějaký požadovaný příkaz. Po uložení platného údaje do crontabu je obvykle již spuštěn program Cron, který běží neustále na pozadí a vykonává požadované příkazy v zadanou dobu.

Tímto způsobem se tedy aplikace k monitorování serverů v síti PlanetLab spouští každý den, pokud je v provozu stanice, na které by měla aplikace pracovat. Zde se dostávám k dalšímu problému, který jsem musel řešit a to ten, že pokud by měla aplikace běžet na mé lokální stanici, musel by být počítač zapnutý ve dne, v noci případně bych jej musel vždy ráno zapínat, večer vypínat a ani jednou na to nezapomenout. Právě proto jsem se rozhodl aplikaci přesunout na některé zařízení PlanetLabu, které bude v provozu neustále.

5.4 Umístění skriptu na server

Při přemisťování aplikace na server sítě PlanetLab bylo důležité nejdříve vyhledat server, kde bude možné spustit Cron a nainstalovat balíček pro funkci příkazu *nslookup*. Již tato akce nebyla příliš jednoduchá, na některých zařízeních totiž nebyly práva pro doinstalování balíčku pro funkci příkazu *nslookup*, na jiných byla zase zakázána editace crontabu a na některých zařízeních nebyla povolena ani jedna možnost. Nakonec byla aplikace umístěna na server *merkur.planetlab.haw-hamburg.de*, který splňoval výše uvedené požadavky. Tento přesun aplikace byl realizován zabezpečeným protokolem SCP.

Poté nastal další problém při prvním spuštění aplikace na serveru, a to ten, že test veškerých zařízení byl spuštěn téměř zároveň a celý běh programu tak trval pouhých několik sekund na rozdíl od testování na lokální stanici, kdy test trval několik minut. To ovšem neslo i úskalí, například některé testy nebyly provedeny, protože na serveru nebyl dostatek volných vláken ke spuštění všech testů. Bylo tedy nutné aplikaci lehce zpomalit, což bylo provedeno jednoduše uspáním programu na 100 ms po každém spuštění jednotlivého testu. Tento úkon je realizován příkazem *sleep 0.1*, který byl vepsán přímo do spouštěcího cyklu aplikace zmíněného v kapitole 5.1. Díky tomu je běh programu stabilizován a přitom trvá pouhých pár minut.

Po vyřešení problému s během programu díky výkonnějšímu hardwaru serveru bylo nutné spouštět aplikaci pravidelně a automaticky. Zde jsem narazil na další problém, protože Cron jsem nebyl schopen spustit na žádném serveru, pokusil jsem se tedy vyřešit problém pomocí funkce *nohup*, která umožňuje nechat spuštěný program i po odpojení vzdáleného SSH přístupu na server. Nakonec stačilo povolit službu Cronu na serveru pomocí příkazu *sudo chkconfig crond on* a následně spustit tuto službu pomocí příkazu *sudo service crond start*. Tímto postupem jsem nakonec dosáhl pravidelného spouštění aplikace každý den v poledne. Po změně na letní čas se aplikace spouští ve 13 hodin každý den. V crontabu je ovšem nastaven čas na 11 hodin dopoledne vzhledem k neaktuálnímu časovému nastavení serveru, který leží v Hamburku, v Německu. Čas na serveru je tedy nastaven přesně o dvě hodiny pozadu. Práva na změnu tohoto nastavení má ovšem pouze správce (tzv. "root").

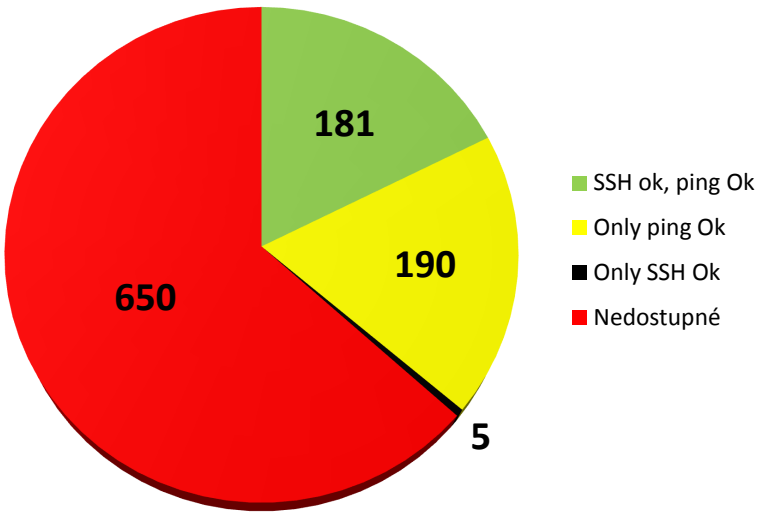
Další problém vznikl při stahování souborů aplikací Mateje Grejtáka. Problém spočíval v tom, že vzdálená aplikace, která měla stahovat naměřené údaje pomocí protokolu SCP, nebyla schopna rozpoznat, zda-li právě neprobíhá měření a mohla stáhnout neúplný výstupní soubor. Tento problém byl vyřešen tím způsobem, že se nejdříve vytvoří záloha výstupního souboru "Result_datum_čas.txt"a do ní je zapisováno. Až po dokončení testování je vytvořen soubor *Result.txt* a záložní soubor s datem a časem je přesunut do složky LOG. Aplikace Mateje Grejtáka pak stahuje informace pouze pokud nalezne v adresáři *Result.txt* bez data a času. Pokud tento soubor nenačte, vyčká 10 minut a zkusí to znovu.

6 ANALÝZA NAMĚŘENÝCH DAT

V této kapitole se dostávám k výsledkům měření dostupnosti služeb ping a SSH na zařízeních sítě PlanetLab, které probíhalo od 11. března do 17. dubna roku 2016. Dále se v této kapitole pojednává o funkci paralelizace vytvořeného programu a následně o úspoře času testování oproti sekvenčnímu běhu programu.

6.1 Dostupnost služeb ping a SSH na zařízeních sítě PlanetLab

Na základě testování odezvy ping a dostupnosti SSH připojení jsem získal údaje o více než 1000 zařízeních z celého světa. Tyto údaje se neustále mění, díky čemuž není možné získat naprosto aktuální statistiku. Nicméně, tyto změny se z pravidla pohybují jen v jednotkách zařízení a dá se tedy říci, že statistika uvedená v grafu na Obr. 6.1 je velmi podobná statistice aktuální. Na grafu je zcela jednoznačně vidět, že více než polovina zařízení je nedostupná. V průběhu tvorby aplikace jsem vypozoroval, že více než 600 zařízení je dlouhodobě nedostupných, čemuž odpovídají i informace na webových stránkách sítě PlabetLab.



Obr. 6.1: Měření dostupnosti služeb 3.4.2016.

6.2 Dlouhodobá statistika dostupnosti služeb v PlanetLabu

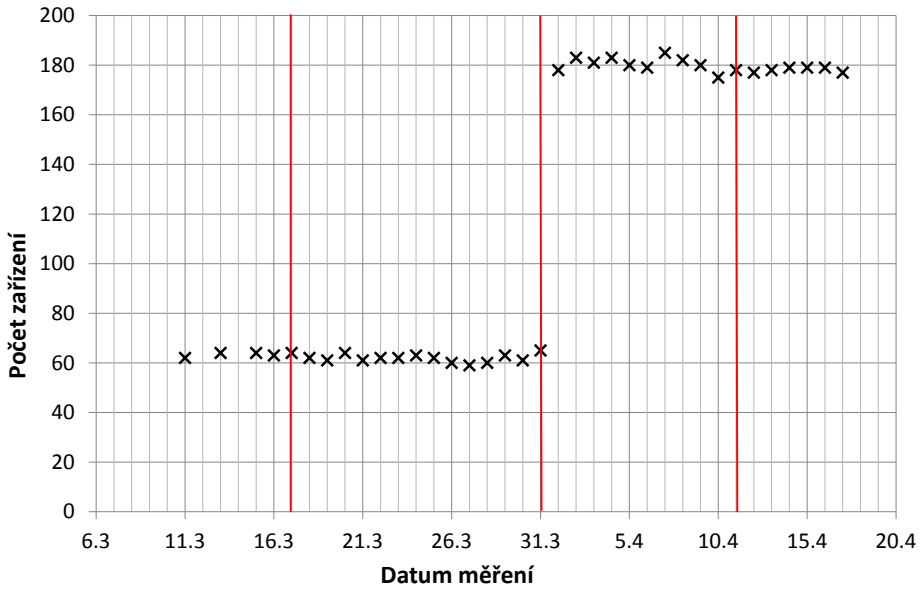
Výsledkem této práce je statistika, která vznikala od 11. března do 17. dubna 2016. Jedná se tedy o výsledky měření dostupnosti služeb v síti PlanetLab za více než měsíc. V každém grafu jsou určitá data, kdy došlo k určitým změnám při měření. Jedná se o data 17. března, 31. března a 11. dubna, která jsou vyznačena v grafech červenou čarou. Dne 17. března došlo k přesunutí aplikace z lokální stanice na PlanetLab server, 31. března došlo ke změně práv uživatele PlanetLabu na službu SSH a 11. dubna došlo ke změně vstupního souboru, kdy se celkový počet stanic změnil z čísla 1026 na 1015. Následky těchto změn jsou lépe popsány dále u příslušných grafů.

Na Obr. 6.2 je zobrazen graf počtu zařízení, která jsou dostupná na obě služby (ping i SSH) v závislosti na čase. Zde je nejlépe vidět skokový nárůst právě 31. března. Jak již bylo zmíněno, ten den došlo k přidání více než 700 zařízení do slice uživatele sítě PlanetLab. Jednoduše řečeno, aplikace se přihlašuje na zařízení PlanetLabu uživatelským účtem a kontroluje tak připojení SSH. Nicméně, tento uživatelský účet měl až do 31. března práva na připojení pouze ke 294 stanicím z celkových 1026 testovaných. Ke zbývajícím 732 stanicím se tedy nebyla aplikace schopna připojit ani v případě, že služba SSH byla dostupná. Tato chyba byla opravena právě 31. března. Díky této nápravě si lze v následujícím grafu všimnout zmíněného skokového nárůstu stanic dostupných na služby ping i SSH z předešlých cca 60 stanic na cca 180 stanic. K této změně došlo již druhý den od přidaní zařízení do slice, tedy 1. dubna. Počet dostupných stanic tedy narostl podle měření přibližně trojnásobně, což přímo odpovídá nyní trojnásobně většímu množství stanic, ke kterým má aplikace právo na přístup.

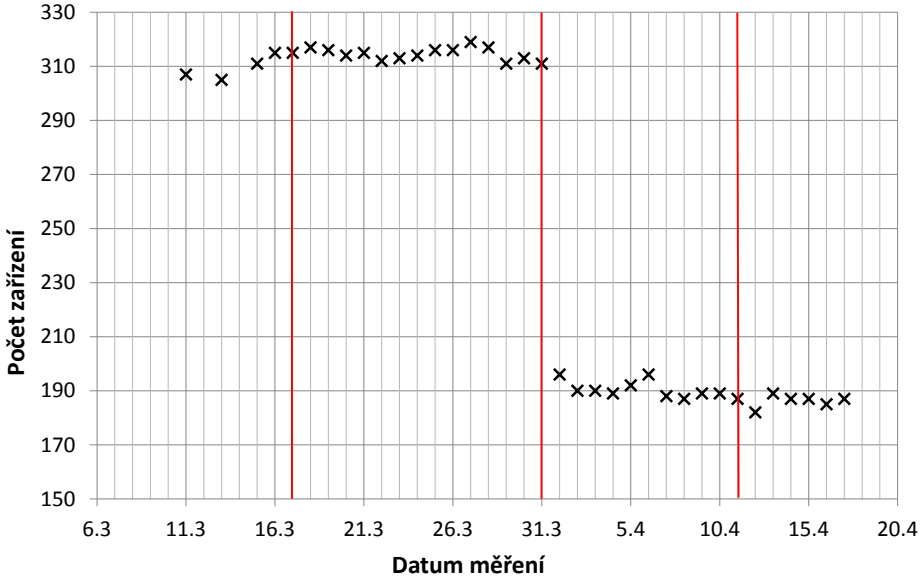
Následující graf zobrazen na Obr. 6.3 představuje diskrétní hodnoty počtů zařízení dostupných pouze na ping v čase. Zde je taktéž velmi dobře vidět stejný problém jako u předcházejícího grafu. Vzhledem k doplnění práv do slice uživatele vzrostl počet zařízení dostupných na SSH, což znamená, že počet zařízení dostupných pouze na ping se zároveň snížil. Tento fakt je znázorněn poklesem počtu zařízení dostupných na ping ale nedostupných na SSH v následujícím grafu.

Další graf, který představuje Obr. 6.4, zobrazuje hodnoty počtů zařízení dostupných pouze na službu SSH v čase. Jedná se tedy o zařízení, která mají službu ping zakázanou a tudíž jsou dostupná jen na službu SSH. Takových zařízení je v síti PlanetLab pouze několik. I přes to si lze všimnout vzestupu počtu dostupných zařízení po 31. březnu. Jedná se o totožný problém, o kterém se hovořilo u Obr. 6.2.

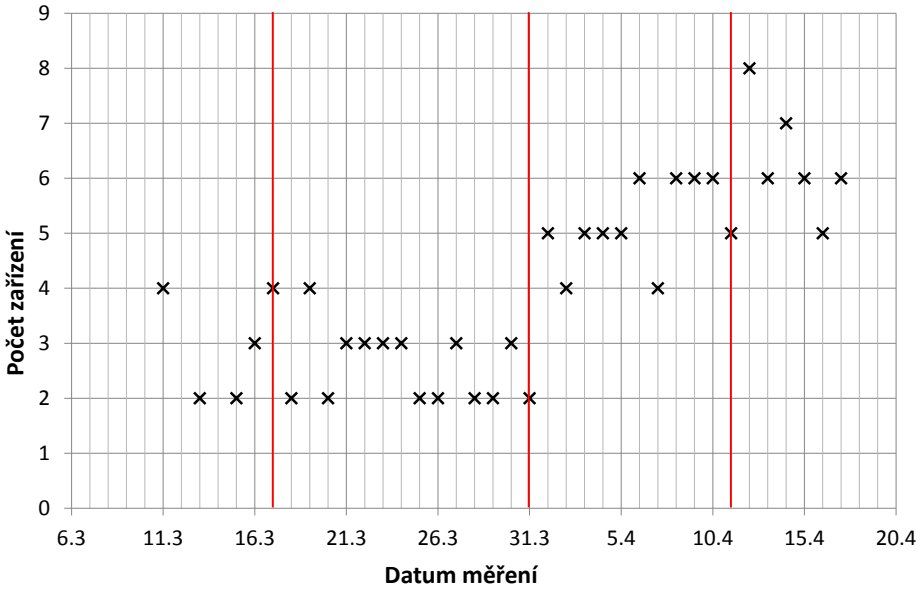
Následující graf, který je zobrazen na Obr. 6.5 představuje průběh počtu nedo-



Obr. 6.2: Graf počtu zařízení dostupných zároveň na ping i SSH.



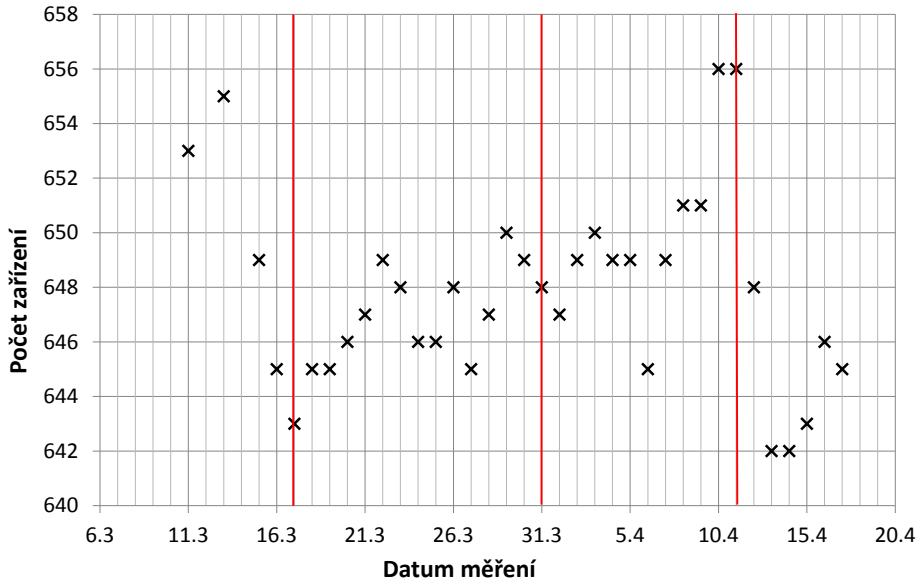
Obr. 6.3: Graf počtu zařízení dostupných pouze na ping.



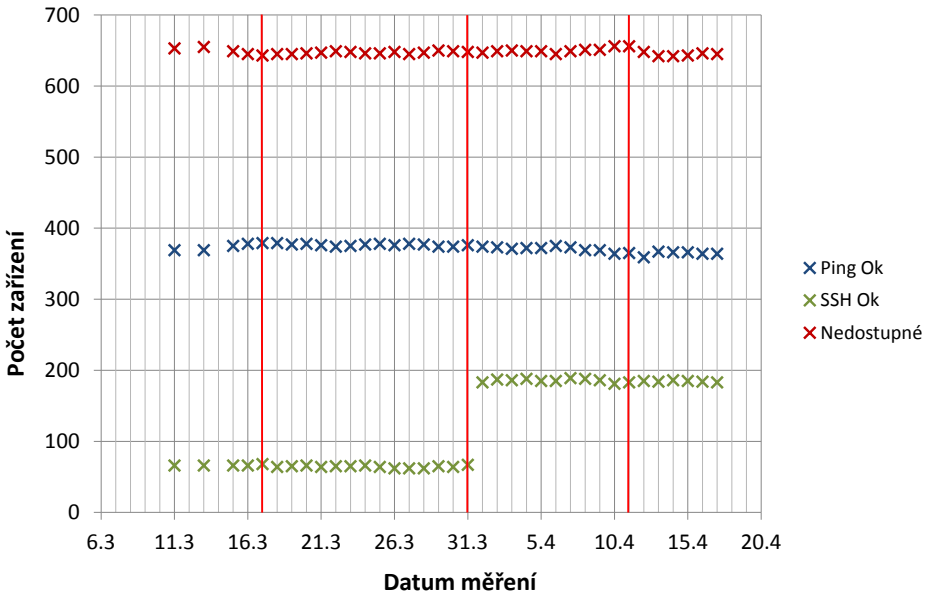
Obr. 6.4: Graf počtu zařízení dostupných pouze na SSH.

stupných zařízení v síti PlanetLab. Tento počet kolísá v intervalu dvaceti zařízení, což je možné vidět v již zmíněném grafu. Dále si můžeme povšimnout skokového poklesu 11. dubna. Tento pokles způsobila změna vstupního souboru, díky čemuž se změnil i celkový počet testovaných stanic z 1026 na 1015. V síti PlanetLab bylo pravděpodobně odstraněno dlouhodobě nefunkčních 9 stanic, což se promítlo do zmíněného grafu.

Na posledním grafu, který je vyobrazen na Obr. 6.6 je možné vidět statistiku všech zařízení PlanetLabu změřenou ve dříve uvedeném intervalu. Modré body odpovídají počtu všech zařízení dostupných v daný den na ping, zelené hodnoty odpovídají všem zařízením dostupným v daný den na SSH a červené body znázorňují počet nedostupných zařízení ani na jednu službu. V tomto grafu je opět vidět strmý nárůst zařízení dostupných na SSH následující den po přidání zařízení do slice uživatele, tedy 1.dubna, tak jako v grafu na Obr. 6.2. Ostatní změny, které se odehrály v průběhu měření nejsou v této celkové statistice příliš patrné.



Obr. 6.5: Graf počtu nedostupných zařízení.



Obr. 6.6: Graf počtu všech zařízení dostupných na ping, na SSH a nedostupných.

6.3 Ověření funkčnosti paralelizace

V průběhu měření byla také ověřena funkčnost paralelizace programu. Nad tímto ověřením bylo uvažováno již při konstrukci programu, a právě proto je ukládán čas počátku a konce testování do datového souboru *metadata.dat*.

Program spouští testy jednotlivých zařízení ve velmi krátkých odstupech pouze díky příkazu *disown* v hlavním cyklu programu, který byl popsán v kapitole 5.1. Je tedy možné nechat pracovat program čistě sekvenčně po zakomentování tohoto příkazu. Této možnosti bylo využito pro měření času testování sekvenčního běhu programu.

Jak je možné vidět v Tab. 6.1, při porovnání sekvenčního a paralelizovaného běhu programu zjistíme, že paralelizací byly ušetřeny v průměru cca 3 hodiny. Ve výsledku tedy paralelizovaná aplikace dokáže zpracovat data o více než 1000 zařízení během několika minut. Dovolím si podotknout, že tato skutečnost nemá vliv na vypočítané statistiky, což bylo ověřeno srovnáním výstupních souborů zmíněných měření.

Toto ověření funkčnosti paralelizace bylo prováděno i na lokální stanici, kde byl rozdíl ještě markantnější. Paralelizovaná aplikace byla schopna otestovat zařízení PlanetLabu během čtyř minut. Pokud jsme aplikaci spustili sekvenčně, test trval přibližně 4 hodiny.

Tab. 6.1: Tabulka ušetřeného času pomocí paralelizace.

	Sekvenční běh	**Paralelní běh**	**Ušetřený čas**
Test paralelizace 23.4.2016	03:07:33	00:02:33	03:05:00
Test paralelizace 8.5.2016	03:08:04	00:04:35	03:03:29
Test paralelizace 9.5.2016	03:03:13	00:06:09	02:57:04
Průměrný čas	03:06:17	00:04:26	03:01:51

7 ZÁVĚR

Na začátku této práce jsem se seznámil s experimentální sítí PlanetLab. Stručně jsem popsal vývoj této experimentální sítě a nejdůležitější projekty, které byly v této síti vypracovány nebo na kterých se ještě stále pracuje. Dále jsem představil služby ping a SSH, kterých se přímo týká praktická část práce, programové prostředí, ve kterém praktická část práce vznikala a paralelizaci procesů, která je použita k optimalizaci praktické části práce.

Dále jsem vytvořil aplikaci, která pravidelně monitoruje servery v Internetu. Tato aplikace nejdříve načte seznam stanic, otestuje dostupnost ping a SSH připojení každé stanice, spočítá statistiky jednotlivých služeb a výsledek vypíše do souboru *Result.txt*. Následně vytvoří další soubory se všemi potřebnými daty a vytvoří z nich zálohu, kterou uloží do složky. Tyto úkony jsou prováděny každý den plně automaticky, díky pravidelnému spouštění aplikace v programu Cron a díky svému umístění na jednom ze serverů PlanetLabu.

V průběhu testování této aplikace jsem zjistil, že je nedostupná více než polovina zařízení sítě PlanetLab ve světě. Na ping a SSH připojení odpovídá pouze asi 180 zařízení z více než 1000 měřených. Pouze na ping pak odpovídá přibližně 190 stanic (celkově na ping odpovídá cca 370 stanic), 4 až 8 stanic odpovídá pouze na SSH připojení, tudíž mají zakázaný ping a asi 650 zařízení neodpovídá vůbec. Tato statistika je lehce zavádějící, protože dostupnost serverů se mění každým okamžikem v závislosti na různých faktorech, jako je například výpadek napájení síťových zařízení nebo jakékoli jiné selhání sítě. Nicméně při testování výše uvedené aplikace jsem zjistil, že více než 600 zařízení je dlouhodobě nedostupných.

LITERATURA

[1] *PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services* [online]. 2007, [cit. 28. 11. 2015]. Dostupné z URL: <https://www.planet-lab.org/>.

[2] IĽKO, P. *Geografická pozice serverů sítě PlanetLab.* Bakalářská práce. Brno: FEKT VUT v Brně, 2014.

[3] KUBIATOWITZ, J., BINDEL, D., CHEN, Y., CZERWINSKI, S., EATON, P., GOELS, D., GUMMANDI, R., RHEA, S., WHEATERSPOON, H., WEMER, W., WELLS, C., ZHAO, B. *OceanStore: An Architecture for Global-Scale Persistent Storage.* Strany 190-201, Cambridge, listopad 2000.

[4] *CoDeeN: A Content Distribution Network for PlanetLab* [online]. [cit. 11. 12. 2015]. Dostupné z URL: <http://codeen.cs.princeton.edu/>.

[5] *DHARMA: Distributed home agent for robust mobile access* [online]. [cit. 11. 12. 2015]. Dostupné z URL: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1498346&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1498346>.

[6] *CORAL: The Coral Content Distribution Network* [online]. [cit. 11. 12. 2015]. Dostupné z URL: <http://www.coralcdn.org/>.

[7] NAVRÁTIL: *PlanetLab - model budoucího Internetu.* Zpravodaj ÚVT MU. ISSN 1212-0901, 2006, roč. XVI, č. 5, s. 1-5. Dostupné z URL: <http://webserver.ics.muni.cz/bulletin/articles/525.html#lit11>.

[8] *Příkazy ipconfig, ping, systeminfo, FTP* [online]. [cit. 11. 12. 2015]. Dostupné z URL: <http://dosms.cz/prikazy-ipconfig-ping-systeminfo-ftp/>.

[9] SSH *Hardens the Secure Shell* [online]. 2005, [cit. 11. 12. 2015]. Dostupné z URL: <http://www.serverwatch.com/news/print.php/3551081>.

[10] *Service Name and Transport Protocol Port Number Registry* [online]. 2015, [cit. 11. 12. 2015]. Dostupné z URL: <http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>.

[11] SSH *Frequently Asked Questions* [online]. 2001, [cit. 11. 12. 2015]. Dostupné z URL: <http://www.snailbook.com/faq/ssh-1-vs-2.auto.html>.

[12] ROSASCO, N., LAROCHELLE, D.: *How and Why More Secure Technologies Succeed in Legacy Markets: Lessons from the Success of SSH* [online]. Dept.

of Computer Science, Univ. of Virginia, [cit. 11. 12. 2015]. Dostupné z URL: <http://www.cs.virginia.edu/~drl7x/sshVsTelnetWeb3.pdf>.

[13] Abc Linuxu: *Seriál: BASH* [online]. Praha: Nitemedia, 2003, [cit. 28. 11. 2015]. Dostupné z URL: <http://www.abclinuxu.cz/serialy/bash>.

[14] *Charakteristika moderních pentií* [online]. Brno: VUT FIT, 2006, [cit. 2. 5. 2016]. Dostupné z URL: <http://www.fit.vutbr.cz/study/courses/PTP/public/texty/pentium03.pdf>.

[15] *Návrh paralelních algoritmů* [online]. Praha: CVUT, 2007, [cit. 2. 5. 2016]. Dostupné z URL: <http://geraldine.fjfi.cvut.cz/~oberhuber/data/hpc/paa/prezentace/10-paralelni-alg.pdf>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

Bash	Bourne again shell
CLI	Příkazový řádek – Command Line Interface
CMU	Carnegie Mellon University
DNS	Domain Name System
DHARMA	Distributed Home Agent for Remote Mobile Access
DHT	Distributed Hash Table
GPS	Global Positioning System
GSSAPI	Generic Security Services Application Program Interface
ICMP	Internet Control Message Protocol
IP	Internet Protocol
Ping	Packet InterNet Groper
IrisNet	Internet-scale Resource Intensive Sensor Network Services
POSIX	Portable Operating System Interface
QoS	Quality of Service
RON	Resilient Overlay Network
RSA	Rivest, Shamir, Adleman
SCP	Secure Copy
SSH	Secure Shell
TCP/IP	Transmission Control Protocol/Internet Protocol
VPN	Virtuální Privátní Síť

SEZNAM PŘÍLOH

A Obsah přiloženého CD **43**

A OBSAH PŘILOŽENÉHO CD

- **xpospi81.pdf:** elektronická verze práce.
- **planetlab.node:** vstupní soubor skriptu a zároveň seznam serverů sítě PlanetLab.
- **Monitoring.sh:** skript na testování dostupnosti sítě PlanetLab.
- **Result.txt:** hlavní výstupní soubor skriptu.
- **metadata.dat:** výstupní datový soubor.
- **pingOk.dat:** výstupní datový soubor se stanicemi dostupnými na ping.
- **pingErr.dat:** výstupní datový soubor se stanicemi nedostupnými na ping.
- **sshOk.dat:** výstupní datový soubor se stanicemi dostupnými na SSH.
- **sshErr.dat:** výstupní datový soubor se stanicemi nedostupnými na SSH.